Merging in Git

Elijah Newren

Overview

- Three-way content merge
- Helpful git commands
- Critiques/Limitations
- Applicability of three-way content merges
- Caveats
- Directory rename detection

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Same file from branch B:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
...
```

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Same file from branch B:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
...
```

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Same file from branch B:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
...
```

Correct merge depends on the version in the merge base:

```
speak_like_a_pirate(arrrgs);
?????
shiver(me.timbers);
```

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Same file from branch B:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
...
```

Correct merge depends on the version in the merge base:

```
speak_like_a_pirate(arrrgs);
?????
shiver(me.timbers);
```

Which we need to know to determine the merge:

```
speak_like_a_pirate(arrrgs);
?????
shiver(me.timbers);
```

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Same file from branch B:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
...
```

Correct merge depends on the version in the merge base:

```
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
```

Which results in the following merge:

```
speak_like_a_pirate(arrrgs);

shiver(me.timbers);
```

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Same file from branch B:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
...
```

Correct merge depends on the version in the merge base:

```
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
```

Which results in the following merge:

```
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
```

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Same file from branch B:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
...
```

Correct merge depends on the version in the merge base:

```
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
```

Which results in the following merge:

```
speak_like_a_pirate(arrrgs);

shiver(me.timbers);
```

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Same file from branch B:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
...
```

Correct merge depends on the version in the merge base:

```
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
```

Which results in the following merge:

```
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
```

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Same file from branch B:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
...
```

Correct merge depends on the version in the merge base:

```
speak_like_a_pirate(arrrgs);
explore_sea(plus, plus);
shiver(me.timbers);
```

Which results in the following merge:

```
speak_like_a_pirate(arrrgs);

shiver(me.timbers);
```

Three-way content merge

File from branch A:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
...
```

Same file from branch B:

```
...
speak_like_a_pirate(arrrgs);
explore_sea(me.love[0]);
shiver(me.timbers);
...
```

Correct merge depends on the version in the merge base:

```
speak_like_a_pirate(arrrgs);
explore_sea(plus, plus);
shiver(me.timbers);
```

Which results in the following merge:

```
speak_like_a_pirate(arrrgs);
<<<<<<< HEAD
explore_sea(aye, matey);
=======
explore_sea(me.love[0]);
>>>>>>> branchB
shiver(me.timbers);
```

Three-way content merge, cont.

git's sha1sum of individual files can be used for a shorthand:

```
  path
1: sha1sum(orig:path)
2: sha1sum(A:path)
3: sha1sum(B:path)
```

Three-way content merge, cont.

git's sha1sum of individual files can be used for a shorthand:

```
  path
1: sha1sum(orig:path)
2: sha1sum(A:path)
3: sha1sum(B:path)
```

For example (using shortened shas here):

```
  angryp.swine-latin
1: 5ca1ab1e
2: f005ba11
3: b0a710ad
```

Three-way content merge, cont.

git's sha1sum of individual files can be used for a shorthand:

```
  path
1: sha1sum(orig:path)
2: sha1sum(A:path)
3: sha1sum(B:path)
```

For example (using shortened shas here):

```
  angryp.swine-latin
1: 5ca1ab1e
2: f005ba11
3: b0a710ad
```

Where the ordering is as follows:

1. merge base
2. HEAD (branch checked out before running merge)
3. other branch (the argument you passed to merge)

Three-way content merge, cont.

git's sha1sum of individual files can be used for a shorthand:

```
  path
1: sha1sum(orig:path)
2: sha1sum(A:path)
3: sha1sum(B:path)
```

For example (using shortened shas here):

```
  angryp.swine-latin
1: 5ca1ab1e
2: f005ba11
3: b0a710ad
```

Where the ordering is as follows:

1. merge base
2. HEAD (branch checked out before running merge)
3. other branch (the argument you passed to merge)

git makes these accessible...

Helpful commands

Getting details about which files are conflicted:

```
$ git ls-files -u
100644 41e3dc22a02a972d0d42 1    angryp.swine-latin
100644 f185132ce93bf3e453b8 2    angryp.swine-latin
100644 b506e78238513afdfbb0 3    angryp.swine-latin
```

Helpful commands

Getting details about which files are conflicted:

```
$ git ls-files -u
100644 41e3dc22a02a972d0d42 1    angryp.swine-latin
100644 f185132ce93bf3e453b8 2    angryp.swine-latin
100644 b506e78238513afdfbb0 3    angryp.swine-latin
```

Viewing other versions:

```
$ git show :stage:filename
$ git show sha1sum
```

Helpful commands

Getting details about which files are conflicted:

```
$ git ls-files -u
100644 41e3dc22a02a972d0d42 1    angryp.swine-latin
100644 f185132ce93bf3e453b8 2    angryp.swine-latin
100644 b506e78238513afdfbb0 3    angryp.swine-latin
```

Viewing other versions:

```
$ git show :stage:filename
$ git show sha1sum
```

Example:

```
$ git show :2:angryp.swine-latin
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
```

Helpful commands

Getting details about which files are conflicted:

```
$ git ls-files -u
100644 41e3dc22a02a972d0d42 1    angryp.swine-latin
100644 f185132ce93bf3e453b8 2    angryp.swine-latin
100644 b506e78238513afdfbb0 3    angryp.swine-latin
```

Diffing against other versions:

```
$ git diff [--base|--ours|--theirs] [filename]
```

Helpful commands

Getting details about which files are conflicted:

```
$ git ls-files -u
100644 41e3dc22a02a972d0d42 1    angryp.swine-latin
100644 f185132ce93bf3e453b8 2    angryp.swine-latin
100644 b506e78238513afdfbb0 3    angryp.swine-latin
```

Diffing against other versions:

```
$ git diff [--base|--ours|--theirs] [filename]
```

Example:

```
$ git diff --base
...
@@ -1,3 +1,7 @@
 speak_like_a_pirate(arrrgs);
-explore_sea(plus, plus);
+<<<<<<< HEAD
+explore_sea(aye, matey);
+=======
+explore_sea(me.love[0]);
+>>>>>>> branchB
 shiver(me.timbers);
```

Helpful commands

Getting details about which files are conflicted:

```
$ git ls-files -u
100644 41e3dc22a02a972d0d42 1    angryp.swine-latin
100644 f185132ce93bf3e453b8 2    angryp.swine-latin
100644 b506e78238513afdfbb0 3    angryp.swine-latin
```

Ovewriting with different versions:

```
$ git checkout [--ours|--theirs] <filename>
$ git checkout [--merge|-m|--conflict=diff3] <filename>
```

Helpful commands

Getting details about which files are conflicted:

```
$ git ls-files -u
100644 41e3dc22a02a972d0d42 1    angryp.swine-latin
100644 f185132ce93bf3e453b8 2    angryp.swine-latin
100644 b506e78238513afdfbb0 3    angryp.swine-latin
```

Ovewriting with different versions:

```
$ git checkout [--ours|--theirs] <filename>
$ git checkout [--merge|-m|--conflict=diff3] <filename>
```

Example:

```
$ git checkout --ours angryp.swine-latin
$ cat angryp.swine-latin
speak_like_a_pirate(arrrgs);
explore_sea(aye, matey);
shiver(me.timbers);
```

Helpful commands

Getting details about which files are conflicted:

```
$ git ls-files -u
100644 41e3dc22a02a972d0d42 1    angryp.swine-latin
100644 f185132ce93bf3e453b8 2    angryp.swine-latin
100644 b506e78238513afdfbb0 3    angryp.swine-latin
```

Ovewriting with different versions:

```
$ git checkout [--ours|--theirs] <filename>
$ git checkout [--merge|-m|--conflict=diff3] <filename>
```

Example:

```
$ git checkout --conflict=diff3 angryp.swine-latin
$ cat angryp.swine-latin
speak_like_a_pirate(arrrgs);
<<<<<<< ours
explore_sea(aye, matey);
||||||| base
explore_sea(plus, plus);
=======
explore_sea(me.love[0]);
>>>>>>> theirs
shiver(me.timbers);
```

Helpful commands

Can look for commits which touched conflicted files:

```
$ git log HEAD...MERGE_HEAD -- \
                    `git ls-files -u | awk {print\$4}`
commit 95d844d711a8ba9fae97 (HEAD, branchA)
Author: Cap'n Blackbeard <black@beard.pirate>
Date: Tue May 22 13:53:03 2018 -0700

    Aye, aye

commit 34fa04c4a1962cf949b3 (branchB)
Author: Cap'n Whitebeard <white@beard.pirate>
Date: Tue May 22 13:52:48 2018 -0700

    Me first love
```

Helpful commands

Can look for commits which touched conflicted files:

```
$ git log HEAD...MERGE_HEAD -- \
                  `git ls-files -u | awk {print\$4}`
commit 95d844d711a8ba9fae97 (HEAD, branchA)
Author: Cap'n Blackbeard <black@beard.pirate>
Date: Tue May 22 13:53:03 2018 -0700

    Aye, aye

commit 34fa04c4a1962cf949b3 (branchB)
Author: Cap'n Whitebeard <white@beard.pirate>
Date: Tue May 22 13:52:48 2018 -0700

    Me first love
```

But there's an equivalent simple shorthand:

```
$ git log --merge
```

Helpful commands

Can look for commits which touched conflicted files:

```
$ git log HEAD...MERGE_HEAD -- \
                    `git ls-files -u | awk {print\$4}`
commit 95d844d711a8ba9fae97 (HEAD, branchA)
Author: Cap'n Blackbeard <black@beard.pirate>
Date: Tue May 22 13:53:03 2018 -0700

    Aye, aye

commit 34fa04c4a1962cf949b3 (branchB)
Author: Cap'n Whitebeard <white@beard.pirate>
Date: Tue May 22 13:52:48 2018 -0700

    Me first love
```

But there's an equivalent simple shorthand:

```
$ git log --merge
```

Which can be handy in combination with other flags, e.g.:

```
$ git log --merge -p --oneline --left-right
```

Helpful commands

Using handy --merge flag to log:

```
$ git log --merge -p --oneline --left-right

< 95d844d (HEAD, branchA) Aye, aye
diff -git a/angryp.swine-latin b/angryp.swine-latin
index 41e3dc2..f185132 100644
-- a/angryp.swine-latin
+++ b/angryp.swine-latin
@@ -1,3 +1,3 @@
 speak_like_a_pirate(arrrgs);
-explore_sea(plus, plus);
+explore_sea(aye, matey);
 shiver(me.timbers);

> 34fa04c (branchB) Me first love
diff -git a/angryp.swine-latin b/angryp.swine-latin
index 41e3dc2..b506e78 100644
-- a/angryp.swine-latin
+++ b/angryp.swine-latin
@@ -1,3 +1,3 @@
 speak_like_a_pirate(arrrgs);
-explore_sea(plus, plus);
+explore_sea(me.love[0]);
 shiver(me.timbers);
```

Helpful commands

Pro-tip: You can ask git to check if there are conflict markers or whitespace errors:

```
$ git diff --check
angryp.swine-latin:2:  leftover conflict marker
angryp.swine-latin:4:  leftover conflict marker
angryp.swine-latin:6:  leftover conflict marker
```

Helpful commands, summarized

Getting details about which files are conflicted:

```
$ git ls-files -u
```

Viewing other versions:

```
$ git show :stage:filename
```

Diffing against other versions:

```
$ git diff [--base|--ours|--theirs] [filename(s)]
```

Ovewriting with different versions:

```
$ git checkout [--ours|--theirs] <filename>
$ git checkout [--merge|-m|--conflict=diff3] <filename>
```

Seeing which commits touched conflicted files:

```
$ git log --merge -p --left-right
```

Checking for remaining conflict markers:

```
$ git diff --check
```

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Files must be “normal” (text)

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Files must be “normal” (text)
  - binaries
  - symlinks
  - submodules

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Files must be “normal” (text)
  - binaries
  - symlinks
  - submodules

Managing: git has some smarts (and some dumbs) for merging these file types; look for “binary” in gitattributes(5).

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Needs file “encoding” to be the same

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Needs file “encoding” to be the same
  - ASCII vs. EBCDIC
  - CR vs. CRLF
  - whitespace normalization
  - unicode normalization
  - other programmatic modifications – indentation, etc.

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Needs file “encoding” to be the same

Managing:

- Strategy options:
  - -Xrenormalize
  - -Xignore-space-change
  - -Xignore-all-space
  - -Xignore-space-at-eol
  - -Xignore-cr-at-eol
  - -Xdiff-algorithm=[patience|minimal|histogram|myers]
- Config options:
  - merge.renormalize
- Important manpages:
  - git-merge(1) (“MERGE STRATEGIES” (recursive), “CONFIGURATION”)
  - gitattributes(5)

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Assumes conflict markers are distinguishable from other text

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Assumes conflict markers are distinguishable from other text
  - What if you're writing documentation about how conflicts work?

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Assumes conflict markers are distinguishable from other text
  - What if you're writing documentation about how conflicts work?

Managing: See "conflict-marker-size" in gitattributes(5)

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Assumes line-based (vs. word-based) diffing

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Assumes line-based (vs. word-based) diffing
  - books
  - documentation
  - articles

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Assumes line-based (vs. word-based) diffing
  - books
  - documentation
  - articles

Managing: Um...don't re-wrap or just suck up extra conflicts?

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Text-based diff ignores (most) semantic content

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Text-based diff ignores (most) semantic content
  - both branches add a new import on the same line
  - one branch adds call to a function, another adds an argument to it

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Text-based diff ignores (most) semantic content
  - both branches add a new import on the same line
  - one branch adds call to a function, another adds an argument to it

File from branch A:

```
...
-def inspect_ship(cannons,
mast)
+def inspect_ship(cannons,
mast, plank)
...
```

Different file from branch B:

```
...
+inspect_ship(MHWCK_045026,
big_pole)
...
```

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Text-based diff ignores (most) semantic content
  - both branches add a new import on the same line
  - one branch adds call to a function, another adds an argument to it

File from branch A:

```
...
-def inspect_ship(cannons,
mast)
+def inspect_ship(cannons,
mast, plank)
...
```

Different file from branch B:

```
...
+inspect_ship(MHWCK_045026,
big_pole)
...
```

Managing: For some cases, extra manual conflict resolution. For other cases, make sure to build and test the merged result, not just the proposed change.

Critiques/Limitations of three-way merge idea

Big hammer: See “custom merge driver” in gitattributes(5)

Critiques/Limitations of three-way merge idea

Big hammer: See “custom merge driver” in gitattributes(5)

- Can define `merge.<driver>.{name,driver}`
- Local-only, though

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Ignores intermediate history

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Ignores intermediate history
  - Same fix applied on both branches. Later, additional fixes were made to the same lines on one branch.

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Ignores intermediate history
  - Same fix applied on both branches. Later, additional fixes were made to the same lines on one branch.
  - A “fix” was applied on both branches (cherry-picked or manually committed in both places), but then reverted on one side because it’s harmful.

Critiques/Limitations of three-way merge idea

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

- Ignores intermediate history
  - Same fix applied on both branches. Later, additional fixes were made to the same lines on one branch.
  - A “fix” was applied on both branches (cherry-picked or manually committed in both places), but then reverted on one side because it’s harmful.

Managing: For some cases, extra manual conflict resolution. For other cases, make sure to build and test the merged result, not just the proposed change.

(If these two cases bother you enough, consider darcs or pijul.)

Applicability of three-way content merges

Under what circumstances is using this algorithm the wrong way to merge three versions of the file?

Applicability of three-way content merges

Under what circumstances is using this algorithm the wrong way to merge ***three versions*** of the file?

Applicability of three-way content merges

Under what circumstances is using this algorithm the wrong way to merge ***three versions*** of ***the*** file?

Applicability of three-way content merges

Under what circumstances is using this algorithm the wrong way to merge ***three versions*** of ***the*** file?

- Can we assume that the same pathname at different points of history implies the content is related? (e.g. could delete a file and add an unrelated one back.)

Applicability of three-way content merges

Under what circumstances is using this algorithm the wrong way to merge ***three versions*** of ***the*** file?

- Can we assume that the same pathname at different points of history implies the content is related? (e.g. could delete a file and add an unrelated one back.)
- Can we assume that the only related content for a file will be found at the same pathname?

Applicability of three-way content merges

Under what circumstances is using this algorithm the wrong way to merge ***three versions*** of ***the*** file?

- Can we assume that the same pathname at different points of history implies the content is related? (e.g. could delete a file and add an unrelated one back.)
- Can we assume that the only related content for a file will be found at the same pathname?
- Can we assume there are three versions of each “file”?

Applicability of three-way content merges

Under what circumstances is using this algorithm the wrong way to merge ***three versions*** of ***the*** file?

- Can we assume that the same pathname at different points of history implies the content is related? (e.g. could delete a file and add an unrelated one back.)
- Can we assume that the only related content for a file will be found at the same pathname?
- Can we assume there are three versions of each “file”?

Let’s dig in, except:

- Git does not (yet?) do break detection (determining whether same-named files are no longer related) for merges.

Applicability of three-way content merges

Under what circumstances is using this algorithm the wrong way to merge ***three versions*** of ***the*** file?

- Can we assume that the same pathname at different points of history implies the content is related? (e.g. could delete a file and add an unrelated one back.)
- Can we assume that the only related content for a file will be found at the same pathname?
- Can we assume there are three versions of each “file”?

Let’s dig in, except:

- Git does not (yet?) do break detection (determining whether same-named files are no longer related) for merges.
- Git does not (yet?) check for movement of blocks of code between files (e.g. a function moved to a different file in one branch, and that function was modified on other branch) for merges. It does detect code movement as part of git blame, assuming appropriate flags are passed.

Applicability of three-way content merges

Assuming previous page problems aren't an issue (i.e.

- Normal files
- Same "encoding"
- No conflict markers already present
- Line-based is okay
- Files with same name are related
- Content doesn't move between filenames
- We have three versions of every file in each revsion

what kinds of errors might Git report as preventing a successful merge?

Applicability of three-way content merges

Assuming previous page problems aren't an issue (i.e.

- Normal files
- Same "encoding"
- No conflict markers already present
- Line-based is okay
- Files with same name are related
- Content doesn't move between filenames
- We have three versions of every file in each revsion

what kinds of errors might Git report as preventing a successful merge?

- Content conflicts

Applicability of three-way content merges

Assuming previous page problems aren't an issue (i.e.

- Normal files
- Same "encoding"
- No conflict markers already present
- Line-based is okay
- Files with same name are related
- Content doesn't move between filenames
- We have three versions of every file in each revsion

what kinds of errors might Git report as preventing a successful merge?

- Content conflicts
- User has changes staged but not yet committed

Applicability of three-way content merges

Assuming previous page problems aren't an issue (i.e.

- Normal files
- Same "encoding"
- No conflict markers already present
- Line-based is okay
- Files with same name are related
- Content doesn't move between filenames
- We have three versions of every file in each revsion

what kinds of errors might Git report as preventing a successful merge?

- Content conflicts
- User has changes staged but not yet committed
- User has unstaged changes that would be overwritten

Applicability of three-way content merges

Assuming previous page problems aren't an issue (i.e.

- Normal files
- Same "encoding"
- No conflict markers already present
- Line-based is okay
- Files with same name are related
- Content doesn't move between filenames
- We have three versions of every file in each revsion

what kinds of errors might Git report as preventing a successful merge?

- Content conflicts
- User has changes staged but not yet committed
- User has unstaged changes that would be overwritten
  - Unstaged delete could come with untracked directory in the way

Applicability of three-way content merges

Assuming we can delete files, what additional kinds of conflicts can we get?

Applicability of three-way content merges

Assuming we can delete files, what additional kinds of conflicts can we get?

- modify/delete

Applicability of three-way content merges

Assuming we can delete files, what additional kinds of conflicts can we get?

- modify/delete
- untracked file in the way of content

Applicability of three-way content merges

Assuming we can delete files, what additional kinds of conflicts can we get?

- modify/delete
- untracked file in the way of content
- untracked directory in the way

Applicability of three-way content merges

Assuming we can also add files, what additional kinds of conflicts can we get?

Applicability of three-way content merges

Assuming we can also add files, what additional kinds of conflicts can we get?

- add/add (both sides add foo)

Applicability of three-way content merges

Assuming we can also add files, what additional kinds of conflicts can we get?

- add/add (both sides add foo)
- directory/file (add foo vs. add foo/bar)

Applicability of three-way content merges

Assuming we can also add files, what additional kinds of conflicts can we get?

- add/add (both sides add foo)
- directory/file (add foo vs. add foo/bar)
- submodule/file

Applicability of three-way content merges

Assuming we can also add files, what additional kinds of conflicts can we get?

- add/add (both sides add foo)
- directory/file (add foo vs. add foo/bar)
- submodule/file
- submodule/directory

Applicability of three-way content merges

Assuming we can also rename files, what additional kinds of conflicts can we get?

Applicability of three-way content merges

Assuming we can also rename files, what additional kinds of conflicts can we get?

- rename/del (A $\Rightarrow$ B vs. delete A)

Applicability of three-way content merges

Assuming we can also rename files, what additional kinds of conflicts can we get?

- rename/del (A $\Rightarrow$ B vs. delete A)
- rename/add (A $\Rightarrow$ B vs. add B)

Applicability of three-way content merges

Assuming we can also rename files, what additional kinds of conflicts can we get?

- rename/del (A $\Rightarrow$ B vs. delete A)
- rename/add (A $\Rightarrow$ B vs. add B)
- rename/add/del (A $\Rightarrow$ B vs. delete A & add different B)

Applicability of three-way content merges

Assuming we can also rename files, what additional kinds of conflicts can we get?

- rename/del (A $\Rightarrow$ B vs. delete A)
- rename/add (A $\Rightarrow$ B vs. add B)
- rename/add/del (A $\Rightarrow$ B vs. delete A & add different B)
- rename/rename(1to2) (A $\Rightarrow$ B vs. A $\Rightarrow$ C)

Applicability of three-way content merges

Assuming we can also rename files, what additional kinds of conflicts can we get?

- rename/del (A $\Rightarrow$ B vs. delete A)
- rename/add (A $\Rightarrow$ B vs. add B)
- rename/add/del (A $\Rightarrow$ B vs. delete A & add different B)
- rename/rename(1to2) (A $\Rightarrow$ B vs. A $\Rightarrow$ C)
- rename/rename(1to2)/add/add (A $\Rightarrow$ B, add C vs. A $\Rightarrow$ C, add B)

Applicability of three-way content merges

Assuming we can also rename files, what additional kinds of conflicts can we get?

- rename/del (A $\Rightarrow$ B vs. delete A)
- rename/add (A $\Rightarrow$ B vs. add B)
- rename/add/del (A $\Rightarrow$ B vs. delete A & add different B)
- rename/rename(1to2) (A $\Rightarrow$ B vs. A $\Rightarrow$ C)
- rename/rename(1to2)/add/add (A $\Rightarrow$ B, add C vs. A $\Rightarrow$ C, add B)
- rename/rename(2to1) (B $\Rightarrow$ A vs. C $\Rightarrow$ A)

Applicability of three-way content merges

Assuming we can also rename files, what additional kinds of conflicts can we get?

- rename/del (A $\Rightarrow$ B vs. delete A)
- rename/add (A $\Rightarrow$ B vs. add B)
- rename/add/del (A $\Rightarrow$ B vs. delete A & add different B)
- rename/rename(1to2) (A $\Rightarrow$ B vs. A $\Rightarrow$ C)
- rename/rename(1to2)/add/add (A $\Rightarrow$ B, add C vs. A $\Rightarrow$ C, add B)
- rename/rename(2to1) (B $\Rightarrow$ A vs. C $\Rightarrow$ A)
- rename/rename(2to1)/del/del (B $\Rightarrow$ A, delete C vs. C $\Rightarrow$ A, delete B)

Applicability of three-way content merges

Assuming we can also rename files, what additional kinds of conflicts can we get?

- rename/del (A $\Rightarrow$ B vs. delete A)
- rename/add (A $\Rightarrow$ B vs. add B)
- rename/add/del (A $\Rightarrow$ B vs. delete A & add different B)
- rename/rename(1to2) (A $\Rightarrow$ B vs. A $\Rightarrow$ C)
- rename/rename(1to2)/add/add (A $\Rightarrow$ B, add C vs. A $\Rightarrow$ C, add B)
- rename/rename(2to1) (B $\Rightarrow$ A vs. C $\Rightarrow$ A)
- rename/rename(2to1)/del/del (B $\Rightarrow$ A, delete C vs. C $\Rightarrow$ A, delete B)
- chains of rename/rename(1to2) and rename/rename(2to1)
  (A $\Rightarrow$ B, C $\Rightarrow$ D, E $\Rightarrow$ F vs. A $\Rightarrow$ F, C $\Rightarrow$ B, E $\Rightarrow$ D)

Applicability of three-way content merges

What if there is no merge base?

Applicability of three-way content merges

What if there is no merge base?

Turns any pair of non-identical files into add/add conflicts.

Applicability of three-way content merges

What if there are multiple merges bases?

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule
- directory/file, directory/submodule, submodule/file

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule
- directory/file, directory/submodule, submodule/file
- rename/delete

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule
- directory/file, directory/submodule, submodule/file
- rename/delete
- rename/add

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule
- directory/file, directory/submodule, submodule/file
- rename/delete
- rename/add
- rename/add/delete

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule
- directory/file, directory/submodule, submodule/file
- rename/delete
- rename/add
- rename/add/delete
- rename/rename(1to2)

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule
- directory/file, directory/submodule, submodule/file
- rename/delete
- rename/add
- rename/add/delete
- rename/rename(1to2)
- rename/rename(1to2)/add/add

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule
- directory/file, directory/submodule, submodule/file
- rename/delete
- rename/add
- rename/add/delete
- rename/rename(1to2)
- rename/rename(1to2)/add/add
- rename/rename(2to1)

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule
- directory/file, directory/submodule, submodule/file
- rename/delete
- rename/add
- rename/add/delete
- rename/rename(1to2)
- rename/rename(1to2)/add/add
- rename/rename(2to1)
- rename/rename(2to1)/delete/delete

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule
- directory/file, directory/submodule, submodule/file
- rename/delete
- rename/add
- rename/add/delete
- rename/rename(1to2)
- rename/rename(1to2)/add/add
- rename/rename(2to1)
- rename/rename(2to1)/delete/delete
- chains of rename/rename(1to2) and rename/rename(2to1)

Applicability of three-way content merges

What if there are multiple merges bases?

Merge the merge bases!

...but do NOT error out with conflicts. Instead, forcibly resolve:

- content conflicts
- modify/delete
- special files: binary, symlink, submodule
- directory/file, directory/submodule, submodule/file
- rename/delete
- rename/add
- rename/add/delete
- rename/rename(1to2)
- rename/rename(1to2)/add/add
- rename/rename(2to1)
- rename/rename(2to1)/delete/delete
- chains of rename/rename(1to2) and rename/rename(2to1)
- mode conflict (file/symlink, submodule/file, exec./plain, etc.)

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions
  - working tree

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions
  - working tree
  - index

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions
  - working tree
  - index
  - recursive-merge-base

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions
  - working tree
  - index
  - recursive-merge-base
- Nested conflict markers

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions
  - working tree
  - index
  - recursive-merge-base
- Nested conflict markers
- Avoiding unnecessary updates

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions
  - working tree
  - index
  - recursive-merge-base
- Nested conflict markers
- Avoiding unnecessary updates
- Dirty changes and untracked files have many ways of getting in the way, especially with current design

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions
  - working tree
  - index
  - recursive-merge-base
- Nested conflict markers
- Avoiding unnecessary updates
- Dirty changes and untracked files have many ways of getting in the way, especially with current design
- Implementation issues that bubble up to the user

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions
  - working tree
  - index
  - recursive-merge-base
- Nested conflict markers
- Avoiding unnecessary updates
- Dirty changes and untracked files have many ways of getting in the way, especially with current design
- Implementation issues that bubble up to the user
  - rename detection heuristics

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions
  - working tree
  - index
  - recursive-merge-base
- Nested conflict markers
- Avoiding unnecessary updates
- Dirty changes and untracked files have many ways of getting in the way, especially with current design
- Implementation issues that bubble up to the user
  - rename detection heuristics
  - diff algorithm (see -Xdiff-algorithm)

Applicability of three-way content merges

Some things you may have not noticed:

- May not readily be able to represent conflicts by adding content to a file at the specified path(s).
- Desired content may not be writable at specified file location
- More than three versions of a file possible (4 or 6)
- Each conflict type has three different resolutions
  - working tree
  - index
  - recursive-merge-base
- Nested conflict markers
- Avoiding unnecessary updates
- Dirty changes and untracked files have many ways of getting in the way, especially with current design
- Implementation issues that bubble up to the user
  - rename detection heuristics
  - diff algorithm (see -Xdiff-algorithm)
  - writing intermediate file-merges to object store immediately

Caveats

Critiques/Limitations of three-way merge idea

- File Content
  - Files must be “normal” (text)
  - Needs file “encoding” to be the same
  - Assumes conflict markers are distinguishable from other text
  - Assumes line-based diffing
  - Text-based diff ignores semantic content
- Ignores intermediate history

Applicability of three-way content merges:

- Merge-base
  - exists
  - unique
- File versions
  - Three versions of each file exist
  - The three versions of the file have same pathname
  - If there are three versions of pathname, they’re related
- Chunks of lines (e.g. functions) do not move between files

Subtle assumptions:

- Path for a file is writable
- No nesting of conflict markers
- Conflicts representable on disk

Implementation limits:

- Dirty changes
- Rename heuristics
- Writing intermediate file-merges to object store immediately

Directory rename detection

More fun:

- Splitting a directory into two (or more)

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming
- rename/rename(2to1), but only due to transitive rename

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming
- rename/rename(2to1), but only due to transitive rename
- Multiply transitive renames

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming
- rename/rename(2to1), but only due to transitive rename
- Multiply transitive renames
- Don’t apply other side’s rename if we did the same rename

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming
- rename/rename(2to1), but only due to transitive rename
- Multiply transitive renames
- Don’t apply other side’s rename if we did the same rename
- Files/directories in the way of subset of to-be-renamed paths

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming
- rename/rename(2to1), but only due to transitive rename
- Multiply transitive renames
- Don’t apply other side’s rename if we did the same rename
- Files/directories in the way of subset of to-be-renamed paths
- Instead of rename, merge a directory with another

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming
- rename/rename(2to1), but only due to transitive rename
- Multiply transitive renames
- Don’t apply other side’s rename if we did the same rename
- Files/directories in the way of subset of to-be-renamed paths
- Instead of rename, merge a directory with another
- N-to-1 directory merge

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming
- rename/rename(2to1), but only due to transitive rename
- Multiply transitive renames
- Don’t apply other side’s rename if we did the same rename
- Files/directories in the way of subset of to-be-renamed paths
- Instead of rename, merge a directory with another
- N-to-1 directory merge
- Nested directory also renamed (and outside of parent)

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming
- rename/rename(2to1), but only due to transitive rename
- Multiply transitive renames
- Don’t apply other side’s rename if we did the same rename
- Files/directories in the way of subset of to-be-renamed paths
- Instead of rename, merge a directory with another
- N-to-1 directory merge
- Nested directory also renamed (and outside of parent)
- Dual directory rename, one into the other’s way

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming
- rename/rename(2to1), but only due to transitive rename
- Multiply transitive renames
- Don’t apply other side’s rename if we did the same rename
- Files/directories in the way of subset of to-be-renamed paths
- Instead of rename, merge a directory with another
- N-to-1 directory merge
- Nested directory also renamed (and outside of parent)
- Dual directory rename, one into the other’s way
- Renaming directory and basenames of each file within it

Directory rename detection

More fun:

- Splitting a directory into two (or more)
- “Partial” directory rename
- Transitive renaming
- rename/rename(2to1), but only due to transitive rename
- Multiply transitive renames
- Don’t apply other side’s rename if we did the same rename
- Files/directories in the way of subset of to-be-renamed paths
- Instead of rename, merge a directory with another
- N-to-1 directory merge
- Nested directory also renamed (and outside of parent)
- Dual directory rename, one into the other’s way
- Renaming directory and basenames of each file within it
- Renamed directory contained subdirs (possibly also renamed), not files